Storing Feature Descriptions as 2-D Trees

Thomas C. Henderson
Dept. of Computer Science
University of Utah
Salt Lake City, Utah 84112

Ernst Triendl
DFVLR Oberpfaffenhofen
8031 Wessling, W. Germany

Abstract

Many methods have been proposed which produce low-level features from digital images, e.g., the raw primal sketch or intrinsic images. However, in some cases the features occur sparsely in the image, and a more efficient storage scheme can be used than a registered array of feature images.

Edges constitute one of the most useful sorts of information for scene analysis. Even though edge responses usually occur sparsely throughout an image, the output from an edge detector in most image analysis systems is itself an image of the same dimensions (but possibly multi-channel) as the original intensity image. Appreciable savings in space and time can be achieved if the full edge descriptions (orientation, radius and likelihood information) are stored as a 2-D tree. This is a binary tree which uses the (x,y) locations of the pixels as keys and splits the data at the median along the key with greatest spread. (I.e., this is a k-d tree for k = 2.)

Introduction

Most machine vision systems include some processing which produces low-level features of the image, for example: edges, surface orientations, illumination, etc. (see Barrow and Tennenbaum 1978). However, some of these features produce very sparse results in terms of the class of images under study. The 2-D tree is proposed as a more efficient representation of such features. The following discussion addresses the problem of edge information, but it applies equally well to many other types of features.

Edge information can serve as the basis for many purposes, including segmentation, shape analysis and image registration. The edge-guided registration of LANDSAT images with other types of imagery and drawings motivates the approach described here. In order to achieve a sub-pixel registration accuracy, full edge descriptions of multi-channel LANDSAT images are computed (see Henderson et al 1981). Edges are described at each pixel as a triple: (orientation, radius, edge likelihood). The orientation and radius locate the edge with respect to the center of the pixel which serves as the origin, and the horizontal and vertical image axes serve as the x-axis and y-axis, respectively. The edge likelihood gives a measure of the similarity of an edge model and the intensities in the neighborhood of the pixel. Thus, the output of the edge detector is a 3-channel image with the same number of pixels in each channel as in the original; the 3 channels give the orientation, radius and edge likelihood of the most likely edge at a given pixel. For a 4-channel LANDSAT image, then, the output is a 12-channel edge description image. Note that many image processing systems provide only the edge likelihood as the output image.

Various data compaction schemes have been proposed for images, the most important being the quad-tree. A quad-tree is a successive subdivision of an image into quadrants, where a non-terminal node represents a non-uniform quadrant and leaf nodes represent a uniform quadrant at some level. This data structure is inappropriate for feature encoding since edges usually occur in thin strips throughout an image; moreover, quad-trees are most useful in conjunction with binary images.

Storage Comparisons

Let us consider now the advantages of storing the edge description as a 2-D tree. (For a description of k-d trees see Friedman et al 1977 .) Suppose that the input image is a k-channel m by n image; let p = mn, and suppose there are ap significant edge responses (i.e., above some likelihood threshold), where $0 < a < 1$, and finally, suppose that each edge description consists of e elements. Then in the standard case, the number of storage elements required is ekp. If the 2-D tree scheme is used, then there are (e + 2)ap storage elements required for each leaf of the tree, as the (x,y) location must be stored in addition to the e channels of edge description; also, there is a $\frac{4apk}{b}$ storage element overhead for the nonterminals in the tree, where b is the bucket size or number of records per leaf and assuming 4 storage elements per non-terminal. Thus, the 2-D tree is more economical whenever

$$a < \frac{e}{\left(\frac{4}{b} + e + 2\right)} .$$

For single channel input image and edge descriptions, the breakeven point is around 14% edge density when the bucket size is 1; below this percentage

of edge responses needing to be stored, the 2-D tree is more economical, while above it, the standard image representation requires less space. When the bucket size is 4, the breakeven point is at 25% edge density. In our application, i.e., edge data associated with a given control point, this efficiency can only become more important as the library of control points is increased.

Processing Comparisons

Typical operations on edge descriptions include channel grouping and spatial grouping. A comparison of the cost of these grouping operations reveals that the 2-D tree representation offers time advantages, too. Consider first the time required to perform channel grouping.

In the usual case, i.e., multi-channel edge description, the number of memory accesses required for channel grouping is kp since at each pixel the edge likelihood channel must be checked for each of the k output edge descriptions. However, if the edge descriptions are organized as a 2-D tree using the (x,y) location in the image as the two keys, then the tree will contain multiple entries with the same (x,y) keys if there are edge responses at the same pixel in more than one channel. Multiple responses can be stored as a linked list associated with a single entry for the (x,y) location. This means that channel grouping can be accomplished by simple tree traversal and the number of memory accesses is $2apk + \frac{4apk}{b}$ for terminals and non-terminals respectively.

Thus, whenever

$$a < \frac{b}{2(2 + b)}$$

then the 2-D tree representation is more efficient. For bucket size of 1, this is around 17% edge responses, and at b = 4, the breakeven point is 33%.

Spatial grouping of edge responses requires computing a weighted average of edge responses occurring in some predefined neighborhood of a given pixel. In the standard array image representation, these neighbors can be found in constant time; thus, if spatial grouping is performed on the m-neighborhood of a pixel, then m memory accesses must be performed for every pixel in the image, i.e., mp memory accesses. For the 2-D tree representation, the entire tree must be traversed, and for each leaf record, all neighbors within the prescribed distance must be found. This gives a number of memory accesses that is proportional to:

$$\log (2ap)2ap + \frac{4ap}{b} .$$

Although no direct comparison is possible, there are two major reasons why the 2-D tree representation will be more efficient than the standard one. First, the 2-D tree only performs spatial grouping at pixels where an edge response occurred and then only accesses those locations in the prescribed neighborhood where edge responses occurred. Second, given the sizes of the representations, it is much more likely that the complete 2-D tree will fit in core memory, whereas with the complete array representation, there will be significant disk I/O overhead.

CONCLUSION

The 2-D tree representation of feature images has been shown to be more efficient under certain conditions than the standard image array representation. In practice, we have found the 2-D tree to offer significant advantages. Obviously, the 2-D tree representation does not offer an immediate visual representation. However, in the case of edge descriptions, this is not a significant disadvantage in that an edge visualization image is computed from the multi-channel edge description image anyway. Thus, this intermediate step applies equally well to the 2-D tree representation.

References

Barrow, H.G. and J.M. Tennenbaum, "Recovering Intrinsic Scene Characteristics from Images," SRI Tech. Rept. 157, April 1978.

Friedman, J.H., J.L. Bentley and R.A. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," ACM Trans. on Math. Soft., Vol.3, No.3, Sept. 1977, pp. 209-226.

Henderson, T.C., E. Triendl and R. Winter, "Edge-Guided Image Registration," 2nd Scand. Conf. On Image Analysis, Helsinki, Finland, June 1981, pp. 106-111.